ALTERION

e-Infrastructure Vision

ALP Testing, January 2001

Acceptable Level of Performance (ALP) Testing

Introduction

Traditionally, performance testing has employed a "stopwatch" paradigm. The test involves executing a task (recalculating a spreadsheet, for example) and recording the time it takes to complete that task. The system that executes the task in least amount of time "wins".

At Alterion, however, we believe that the results of stopwatch tests can be misleading – under certain circumstances, use of the stopwatch paradigm can persuade the user to select a "winning" system that is *worse* than the "losing" system.

To provide users with more meaningful results, Alterion introduces the concept of Acceptable Level of Performance (ALP) testing. The focus of ALP testing is on how a system handles complex problems. In particular, ALP testing seeks to establish the levels of complexity a system can handle before its performance becomes unacceptably slow.

ALP results are meaningful – if one system is ranked higher than another, it is because that system can handle more data while still being acceptably fast. Users will be able to use the higher-ranked system longer, even as data needs grow with time.

The Problem with Stopwatch Testing

To create a traditional stopwatch test, a developer defines a task such as "recalculate a spreadsheet with 10,000 formulas." This task is encoded into a script and executed on a variety of systems. The test records the time needed to execute the test; systems with faster times are said to be better.

Unfortunately, the selection and execution of a single common task has a couple of drawbacks:

- A small task can be too small. As machines get faster, most ordinary tasks take place virtually instantaneously. The faster the task the more difficult it is to time – the resolution of the timer becomes a significant issue. One solution is to make the task larger (this approach has flaws as discussed below). A second approach is to execute the task a large number of times (for example, 100), measure the larger time, and divide it by the number of cycles (100) to get a per-task time. Using this second approach provides a precise, but meaningless number – from the user's perspective, a task taking 0.001 seconds is no different than a task taking 0.02 seconds, despite the fact that the latter result is 20 times slower.

About Alterion

Alterion provides independent consulting and testing services to government and corporate information technology decision-makers.

Alterion specializes in a variety of client-oriented testing and analysis services. These services include:

- Acquisition Support
- Analysis and Evaluation Services
- E-Commerce Analysis
- Independent Verification and Validation
- Benchmark Testing

The Alterion Difference

Today's IT decision makers have numerous choices when it comes to IT support services. But few of those choices offer both the technical excellence and the independence of Alterion. True Independent Verification and Validation Services (IV&V) cannot come from a company that sells advice on one hand and systems on the other. Alterion is different because its only business is the evaluation and analysis of IT systems.

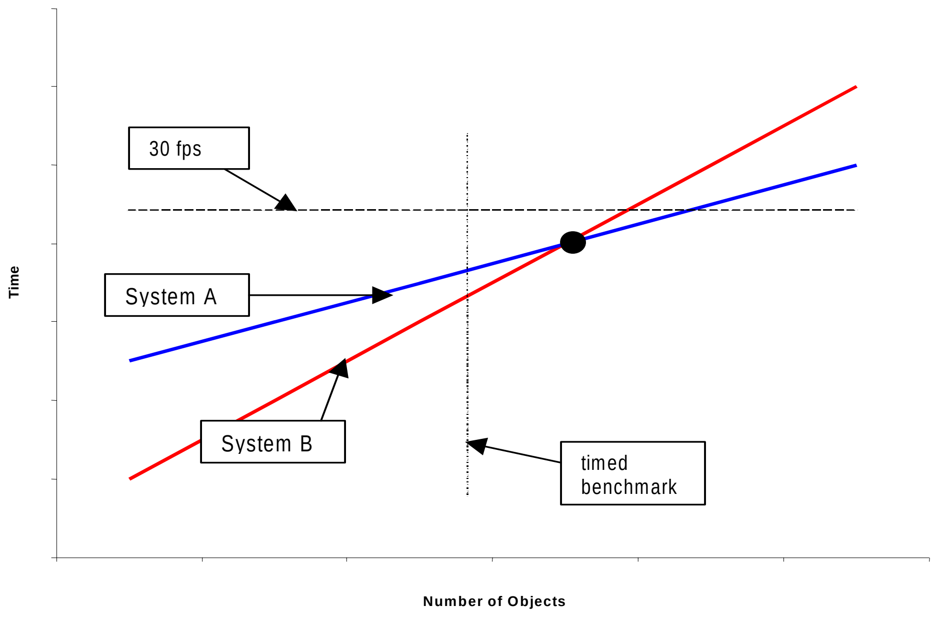
Alterion sells neither hardware nor software. It offers only technical evaluations of hardware and software systems. While government and corporate IT planners can find opinions about products in many places, only Alterion provides the cold, hard facts about how systems will perform in the client's environment. Using a rational, methodological approach, Alterion's technical staff gathers definable and defensible facts to support the best IT decisions.

- A large task can be *too* large. Many activities (such as the graphical display of information) can vary greatly from system to system. The selection of a task that is large enough to thoroughly exercise all machines is unlikely to be attempted on most. Simply put, why test a machine with a task no one will ever want it to do?

The fundamental problem with stopwatch testing, however, relates to the measurement of small activities and which machine characteristics affect performance.

Many activities can be broken up into two components: setup time and load-related time. For example, to display a number of triangles on a screen, there is a fixed setup time (clear the screen, prepare the triangle-processor) and a load, or task, related time (draw each triangle). The fixed time is the same for 100 triangles as it is for 1000 while the load time is proportionate to the size of the task.

Now consider two different graphics processors. One may have a relatively slow setup time but be very fast when drawing triangles. The other may have a very short setup time but a slower triangle-rendering engine. The graph below shows how long it would take each of these systems to draw scenes as a function of how many triangles are being drawn. The first engine, System A, has a slower setup time, while System B has a fast setup time. At low triangle counts, System B is much faster. At the one point where the lines intersect, both systems operate at the same speed. In more complex scenes, System A is faster: Although initially slower, its quicker per-triangle rendering allows it to catch up with, and ultimately exceed the other system.



The vertical dotted line represents a typical timed benchmark for a given scene. The test measures the time needed to display a defined scene. The results analysis considers smaller times to be better, since this means a higher frame rate, and better visual display of information. In this example case, the test decides that System B – with the faster setup time – is better. It draws the given scene in less time.

But at what point is faster not necessarily better? The human vision system reaches maximum perception when a display system reaches about 60 frames per second (FPS) (a display time of around 0.0167 seconds per frame.) Users simply cannot perceive the difference between a system running at 70 FPS and one running at 90 FPS. Only when rates fall below 60 FPS, will the user begin to notice a jerkiness to the image; when the rates drops below 20 FPS or so, the image starts looking quite unusable.

Timed benchmarks don't capture this phenomenon. They are insensitive to the needs of the user. Timed tests may indicate that a machine with 40FPS performance is best if all of the other machines have lower performance. But the user will not be satisfied with the results.

Tests Meaningful to the User – Acceptable Level of Performance Testing

Using the ALP approach to testing the test designer selects an acceptable level of performance. Using the video graphics example from above, an acceptable "level" is established at 30 frames per second. The performance goal is to then discover how complex a scene a system can draw while maintaining this level of acceptable performance.

In the video graphics example, the horizontal line of dashes represents the predefined limit of acceptable performance. (This line is set at 0.033 seconds, the time needed for a single frame at 30 fps.) The chart indicates that System B can handle fewer triangles at 30 fps than System A. In other words, System A can display more complex scenes, with greater detail, while maintaining the appearance of continuous motion. For a user, System A is better.

The measure of the usefulness of a benchmark is how well the test – not the results – applies to the user's needs. If a test's model can be directly applied to what a user does, the results are meaningful to the user. If the model is not relevant, the benchmark is less than useful.

As was just shown, an improperly chosen load in a stopwatch benchmark can easily lead to unsatisfactory results. If the load does not represent the type of load that a user generates, the test may pick a wrong "best system." If the load *is* representative of what the user does, but the test is too short in duration, the results are also open to being misinterpreted. The "faster" and "slower" machines appear to operate instantaneously since the test design does not reflect the true (over time and load) performance of the system.

ALP benchmarking avoids this problem by not choosing a finite load. Instead, analysts set a user-oriented "fast enough" point and the testing determines how much of a load the system can handle before falling below this point. If the result is below the pre-set "fast enough" point, the system will appear slow. If the result is above the real user's typical load, the system will appear sufficiently fast. Further, the difference between the real user's typical load and the measured load tells the user how much the actual load can grow before the system appears to slow down.

This is the core of ALP testing: The selection and execution of tests centered on the needs of the user. Primarily, the user is concerned with whether or not the machine responds in what is perceived to be "fast enough". By choosing a task that represents common user functions and a time goal representative of what a user considers acceptable, ALP testing is more effective at measuring the ability of a system to meet and/or exceed the user's needs.

Examples of ALP Testing

Most computational activities readily fall into the ALP model. This section will describe three activities along with how ALP measures the objects being tested. Each will use a common approach to designing the ALP tests.

- Analysts chose a common, scalable task. The task is one in which a large amount of work is performable between moments of user interaction. Examples include such activities as recalculating a spreadsheet or rotating an image of an object. In each of these examples a single request may involve a small amount of work (for a small spreadsheet or simple object) or a large amount of work (for a large spreadsheet or complex object). Note that tasks such as simple typing are not chosen since the user interaction component is quite frequent and very little work is performed between interaction moments. Also, test designers typically do not choose certain tasks such as file load/save operations as these tasks are fairly uncommon during the majority of a typical work session.
- The test designers choose a measurable target. In the spreadsheet example, recalculation should be fast enough to be completed before the user would attempt some other operation. Typical times might be between 0.25 and 1.0 seconds. Graphic visualizations demand a higher criteria, needing to be refreshed 60 times a second (0.0167 seconds per refresh) to maintain a sense of continuity. Users tend to be a little more flexible for chart redrawing, so times between 1 and 5 seconds might be considered acceptable.
- The designers must sharply define the task environment and vary the size of the operands while repeatedly executing the task. The spreadsheet being recalculated, for example, is made larger and larger until the time needed to complete the task exceeds the target time for that task. When the time limit has been reached, the load causing this limit is reported as the maximal load the system can process in an acceptable amount of time.

Application Performance

Common applications represent where the users spend most of their time, and include products such as Microsoft Word, Microsoft Excel, Adobe Photoshop, and AutoDesk AutoCAD. These applications tend to be user-activity intensive, and most operations take place on very small data items (that is, adding a line to a drawing or typing a character). Certain operations, though, can cause significant delays. For example, a recalculation of a large (40,000-cell spreadsheet) may take a significant amount of time. Operations of the latter sort are selected for ALP testing; any operation that affects only small data segments and is very fast is considered to be unimportant. Most users consider minor delays (under a second) acceptable for admittedly large operations, so ALP time goals in the 1-second range (0.25 to 5 seconds) are commonly chosen.

Graphics System Performance

In a graphics environment, continuity of the scene during viewing is of paramount importance. Therefore, the measure of "acceptable" is the ability of the graphics system to maintain a fairly high frame rate while displaying increasingly complex scenes.

Today, many graphics benchmarks measure the frame rates while rendering known scenes within games such as Quake. However, these benchmarks are limited to the scene complexities of the given samples. Once 60 fps has been surpassed, the displays are essentially continuous.

Instead, an ALP graphics benchmark establishes a scene environment, then goes on to render scenes of increasing complexity. When 60 fps can no longer be attained, the scene has become too complex. The ALP benchmark indicates, for the given environment, how much more "scene complexity" one system can handle over another.

In the graphics arena, different ALP goals are expected for different target audiences. Game players, a driving force in the high end shaded 3D graphics arena, demand a system to render complex scenes at very high frame rates. CAD designers on the other hand, will frequently be viewing line-drawn or flat-shaded images for shorter periods. They need reasonably fluid motion, but can tolerate a higher level of jerkiness (frame rates as low as 15 fps may be considered acceptable).

Server Performance

The testing of server performance probably represents the most significant reason to move from the stopwatch to the ALP model. In a typical stopwatch benchmark, a defined, large series of transactions are thrown at a server environment, measuring the overall time needed to satisfy the requests and providing a time-per-request figure. Unfortunately, this measurement suffers if either the server is faster than the client base, or the server is overloaded.

If the server is faster than the client base, it is certainly possible to measure per-query response to get *some* measure of server performance, but this measurement doesn't have much weight since it only measures how fast the server is when *all* resources are fully available. If the server is overloaded, the stopwatch model does not indicate *at what point* the server becomes overloaded. This "point of overload" is the primary concern to designers configuring servers: An "under-loaded" server will provide reasonably constant per-request performance for its clients, but response times may well begin to decay when resources are fully utilized.

ALP-style server benchmarks meet these various response time issues head-on. They establish the expected profile of a variety of typical clients – each one representing single a unit of load – and increase the number of clients – the load – until performance degrades beyond the point of acceptability.

(It should be noted that more complex benchmarks focused on individual server styles – such as TPC for backend database and SPECWeb for web servers – already use a similar model which measures results in terms of transactions or number of clients supportable within a given response characteristic. These benchmarks establish a server environment, and, with a set of client systems emulating a very large client load, create increasingly higher loads until the server environment can no longer maintain a set throughput level.)

When is ALP Testing Not Appropriate?

The ALP model is no panacea for benchmark testing – not all objects should be forced into the ALP framework. ALP delays are defined as being "a delay that will not affect the user's work state," so activities such as printing files or transferring data across modems falls outside the ALP paradigm.

With printer and modem technologies, the user *expects* a measurable delay in their workflow. Also, these systems are fundamentally sequential in nature: the objects being manipulated are large, single entities that appear to be (and are) handled one piece at a time. For these cases, the traditional stopwatch approach works well enough.

Conclusions

Acceptable Level of Performance testing is, in many cases, the most appropriate approach to use in evaluating a system's performance. By focusing on the user's experience instead of arbitrary metrics, ALP testing yields results that are appropriate, meaningful, and useful.

Contact Alterion

If you would like to contact Alterion to discuss how Alterion can help you evaluate your IT products or services, or just to discuss the topic of this paper, we can be reached through any of the following means:

Address:

Alterion
625 Ridge Pike
Building E, Suite 202
Conshohocken, PA 19428

Email:

General Information requests:
info@alterion.com

Sales information requests:
sales@alterion.com

World Wide Web:

www.alterion.com

Voice:

(610) 832-9450

Fax:

(610) 832-8399